AFRL-IF-RS-TR-2001-231
Final Technical Report
October 2001

A NEW INFRASTRUCTURE FOR EVOLUTIONARY DESIGN AND IMPLEMENTATION

Columbia University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. E101/05

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2001-231 has been reviewed and is approved for publication.

APPROVED: James R. Milligan

JAMES R. MILLIGAN
Project Engineer

Michael L Talbert

FOR THE DIRECTOR: MICHAEL L. TALBERT, Major, USAF
Information Technology Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFTD, 525 Brooks Rd, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

A NEW INFRASTRUCTURE FOR EVOLUTIONARY DESIGN AND IMPLEMENTATION

Gail Kaiser

Contractor: Columbia University
Contract Number: F30602-97-2-0022
Effective Date of Contract: 1 December 1996
Contract Expiration Date: 31 May 2001

Short Title of Work: A New infrastructure for Evolutionary Design and Implementation
Period of Work Covered: Dec 96 – May 01

Principal Investigator: Gail Kaiser
Phone: (212) 939-7081
AFRL Project Engineer: James R. Milligan
Phone: (315) 330-3013

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by James R. Milligan, AFRL/IFTD, 525 Brooks Rd, Rome, NY, 13441-4505.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Oct 01	Final Dec 96 - May 01

4. TITLE AND SUBTITLE
A NEW INFRASTRUCTURE FOR EVOLUTIONARY DESIGN AND IMPLEMENTATION

5. FUNDING NUMBERS
C - F30602-97-2-0022
PE - 62301E
PR - E101
TA - 01
WU - 01

6. AUTHOR(S)
Gail Kaiser

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Columbia University
Dept of Computer Science
500 West 120th Street
New York, NY 10027

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, VA 22203-1714

AFRL/IFTD
525 Brooks Rd
Rome, NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2001-231

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: James R. Milligan, IFTD, 315-330-3013

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Columbia University developed and packaged technologies intended to reduce the time and costs of maintaining large legacy software systems and increase the efficiency and quality of changes to those systems. Columbia produced frameworks, middleware, and components that can be combined with other software development environment and tool products. The focus was on facilities that help software designers, developers, maintainers, users, their managers and other stakeholders to efficiently find, organize, analyze, synthesize and exploit the design rationale and other information they need in large, heterogeneous, disconnected repositories of formal and informal materials describing complex software systems and their development processes. Columbia was particularly concerned with intra-team and inter-team collaboration services, process/workflow, and information management. Their prototype systems enabled users to continually customize and configure the group information spaces of software development environments to optimize them to the software requirements and evolutionary trajectory of immediate concern.

14. SUBJECT TERMS
Avionics Software, Architecture and Design, Software Understanding, Testing and Certification

15. NUMBER OF PAGES
20

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Standard Form 298 (Rev. 2-89) (EG)
Prescribed by ANSI Std. 239.18
Designed using Perform Pro, WHS/DIOR, Oct 94

Table of Contents

1. Objectives

Columbia University proposed to develop and package technologies intended to reduce the time and costs of maintaining large legacy software systems and increase the efficiency and quality of changes to those systems. Columbia investigated frameworks, middleware and components that can be combined with other EDCS results and/or COTS products in software development environments and tools. Their focus was on facilities that help software designers, developers, maintainers, users, their managers and other stakeholders to efficiently find, organize, analyze, synthesize and exploit the design rationale and other information they need in large, heterogeneous, disconnected repositories of formal and informal materials describing complex software systems and their development processes. Columbia was particularly concerned with intra-team and inter-team collaboration services, process/workflow, and information management. Their resulting prototype systems enable users to continually customize and (re)configure the group production information spaces of software development environments to optimize them to the software requirements and evolutionary trajectory of immediate concern. Their approach supports fine-grained, frequent, incremental interactions among the individuals and teams participating in large-scale long-lived software engineering projects that may be geographically and temporally dispersed across multiple autonomous organizations.

2. Technical Results

Prof. Kaiser, the PI, has investigated software process modeling and enactment since 1986, initially in the Marvel project. In the early 1990s in the Oz project, her lab introduced cross-organizational processes operating over the Internet. Oz enabled the software development team and other stakeholders to be geographically, temporally and/or organizationally dispersed. OzWeb, introduced early in the EDCS effort, added integration of Web and other external information resources whereas Oz and Marvel had assumed all project materials to reside in their native objectbases. OzWeb's plugin services and tools were accessible via conventional Web browsers, HTTP proxies and Java GUIs, improving dramatically on Marvel's and Oz's X11 Windows XView/Motif user interface clients. The successive prototype frameworks were used on a daily basis in-house to maintain, deploy and monitor their own components and APIs.

The new process technology developed by Columbia under this cooperative agreement was broadly based on this decade+ of research on and experimentation with architecting and using software development processes targeted to Internet/Web middleware and applications, but reflects a major departure from previous directions. In particular, contemporaneous process and workflow systems were often too rigid for open-ended creative intellectual work, unable to rapidly adapt either the models or the enactment to situational context and/or user role. On the other hand, the process/workflow ideal implies a flexible mechanism for composition and coordination of information system components as well as human participants.

Mobile workflow agents, called *Worklets*, address both the problems and the promise: Worklets might be constructed or parameterized on the fly by a human or a program, then transmitted from component host to host through a "meta-workflow" - a dynamically determined routing pattern reactive to the latest host's circumstances and surroundings as well as past and planned trajectories. Workflow typically involves actions performed on data, or perhaps interactions among humans concerned with implicit data "resident" in the humans' memories. But here the "work" focuses on (re)customizing the host's configuration - loosely construed, including, e.g., schemata, lock tables, authorization capabilities, event subscriptions, even host machine registry. And, as its name implies, worklets can update the process model(s) of a workflow management system. In the degenerate case of the usual data, a worklet is simply a workflow snippet whose semantics are dependent on the host's interpretation of its directives. [Note that by host is meant a particular information system component, not necessarily the entire machine or operating system platform.]

Each worklet is a small mobile program, like the various web agents a combination mobile agent and smart RPC, but in this case potentially including workflow-like rules, as well as imperative code for host-context exploration/instantiation. When worklets manipulate the configuration model(s) of a middleware service or a complex document, the level of dynamism is limited only by the capabilities of the host (as is or wrapped). For example, in the case where the host is a database management system and the worklet initiates changes to its schema, that "(re)configuration" might immediately evolve all data, upgrade data as it happens to be accessed, apply only to new data, or become effective only after a long off-line process, depending on the database system's innate functionality. However, the "configuration" implied by the database's contents could usually be modified on the fly as worklets arrive or the triggering conditions of already-local worklets become satisfied.

As another example, worklets might define part of or modify the workflow definition being enacted by a conventional workflow management tool, inserting their bodies into the model or matching against existing tasks to be adapted or removed. Whether or not a newly modified process model applies to any in-progress process steps, the current or following spiral iteration, or only to the "next" instance is generally limited by the capabilities of the base workflow management system. Unless, of course, the worklet enacts a workflow fragment on its own, which is where the greatest performance/functionality gains and flexibility can be achieved. Any part of an intelligent document could be treated as a configuration model to be upgraded by the worklet, e.g., to tailor and install its components in a distributed enterprise setting.

A host-specific worklet adaptor must be constructed for each anticipated host system or component, and is attached to that host. Obviously, construction of such adaptors is plausible only if the host provides an API or extension language, can reasonably be wrapped, or if its source code is available and the adaptor builder is willing and able to plunge into it. Generally, the adaptor builder must have expert-level understanding of the host and the capabilities it exports. Ideally, however, the worklet writer should have no

need to understand any particular host, other than the generic category of potential hosts (e.g., workflow automation tools) likely to receive the worklet.

Prof. Kaiser's Oz process-centered software development environment framework was perfectly poised to exploit the emergence of the World Wide Web in the mid-1990's for additional reasons beyond the opportunity for mobile workflow. Her lab's proof-of-concept realization of OzWeb added a new kind of built-in object base class, `WebObject`, to the native object management system. In addition to directly storing the object content, WebObjects also contained a URL pointing to that content's "home" at any website on the Internet (or intranet). The local content was treated as a cache, with the remote website queried via HTTP conditional GET - which retrieves the web entity only if it has changed more recently than the cached copy. Users could access WebObjects either through the native X11 Windows client originally constructed for Oz, or through any web browser configured to use their HTTP proxy.

When the browser requested a URL that matched a WebObject, it was retrieved from the OzWeb server along with added-on HTML showing the attributes, relationships, etc. imposed on the entity within OzWeb. But when the browser requested any other URL, not currently known to OzWeb, the proxy forwarded the request to the appropriate external website. In this case the user interface only added on a frame giving the user the option of immediately adding that web entity to the OzWeb objectbase. OzWeb also supported HTTP PUT, for updating backend websites containing in-progress project materials.

Although sufficient to support her lab's own software development, this approach didn't scale very well as they attempted to add on other kinds of Internet and proprietary protocols, besides WebObjects/HTTP. This is not very surprising: the OzWeb code was essentially legacy code that had far outlived its origins in Prof. Kaiser's 1986 Marvel design. Its over 300k source code lines had been added to or modified by about fifty students, included some code written a decade earlier, and was still based on the mid-1980's Unix/C model. OzWeb was ready to retire. Prof. Kaiser's lab started over again, with a new design and architecture, coding in Java, and targeting the Windows NT platform - to produce *Xanth*. They also further componentized the old OzWeb facilities, which had been in progress since the later versions of Marvel, with all the new components also written in Java. For instance, the original Pern transaction manager component was redesigned and reimplemented from scratch as *JPernLite*. The *Rivendell* tool service was integrated as a mandatory component of University of California at Irvine's Chimera open hypermedia system, also supported by EDCS, to launch its viewers.

Xanth neatly partitioned data access modules (DAMs) for accessing arbitrary backend data sources through their native protocols, presentation access modules (PAMs) for appearing to arbitrary front-end user interface and tool clients as their native servers, and service access modules (SAMs) for inserting hyperlinking, annotation, user authorization, workflow, transaction management, etc. services wrapped around PAM and DAM operations. The SAMs were connected to each other and the DAMs and PAMs via a

novel event bus, called the *Groupspace Controller*, which not only propagated notification events but also supported request events that could be vetoed by any service so registered. Veto capability is needed to realize workflow constraints, transaction all-or-nothing guarantees, etc. The conventional event notification after the fact of a prohibited activity is obviously too late. Many events (e.g., sending email, printing) simply cannot be undone or fully compensated, and those that can incur substantial overhead that is unnecessary if the architecture had allowed for them to be prevented in the first place.

Xanth enabled to reimplement OzWeb effectively and efficiently, in about 50k lines of Java code, through a fully scalable architecture. Columbia then easily incorporated a variety of backend data sources like CVS source code repositories, NNTP newsgroups, Ical group calendar managers, and so on. Prof. Kaiser's lab also developed a variety of Web-oriented user interfaces for Xanth, moving away from relatively limited HTML to try browser-resident applets and host-installed apps, as well as legacy clients, e.g., Chimera linkbase viewers.

But none of these user interfaces were truly satisfactory. Like other software development environment researchers and commercial developers, they were using single-user styles of user interface as clients for an inherently collaborative multi-user system. They realized then that they needed to develop groupviews: a user interface style whose core centers on collaboration. The best examples that could be found of such user interfaces were in extremely popular on-line games and socializing forums: 3D virtual worlds and MUDs. These forums are actively used by the general populace, schoolage children to the elderly, with no formal computer science training and often not even computer literacy training. Users pick it up through intuition from the physical world counterpart and informal peer help.

These insights led to Prof. Kaiser's *CHIME* (Columbia Hypermedia IMmersion Environment) project, initiated towards the end of the EDCS effort. One of the project's most deeply seated tenets is to leverage success, such as achieved by 3D multi-player games and multi-user domains (MUDs), in devising usable, useful and used groupviews. Systems constructed using the CHIME infrastructure present their users with a 3D depiction of hypermedia and/or other information resources. Users visualize, and their avatars operate within, a collaborative virtual environment based on some metaphor selected to aid their intuition in understanding and/or utilizing the information of interest or relevant to the task at hand. Users "see" and interact with each other, when in close [virtual] proximity, as well as with the encompassing information space. Actions meaningful within the metaphor are mapped to operations appropriate for the information domain, such as invoking external tools, running queries or viewing documents.

In the proof-of-concept implementation of CHIME, the base data from one or more sources is first mapped to extensible subtypes of the generic components: containers, connectors, components and behaviors, in a virtual model environment (VEM). This includes specifying relationships (connection and containment) among entities from the same and different sources, which might be imposed by the application rather than

inherent in the data. A VEM is then mapped to extensible subtypes of multi-user domain facilities like rooms, doors or hallways between rooms, furnishings, object manipulations, and so on. These are in turn rendered and activated according to the chosen 3D theme world "plugin", which can be dynamically loaded into the generic theme manager at run-time and thence transmitted to the user clients. The same VEM can be mapped simultaneously to multiple theme managers, which can be useful for debugging, administration and system monitoring (although it would probably too confusing for members of the same collaborative team to operate within significantly different "views").

Thus an e-commerce web site peddling computer hardware might look and feel like an on-screen CompUSA; a digital library might be illustrated as, indeed, a library. Application domains without obvious physical counterparts might choose more whimsical themes. For example, a software development environment for an open-source system might map each source code package to a room on the Starship Enterprise, with the "main" subprogram represented by the bridge, amateur programmers proposing a modification could beam aboard, and so forth. Note these are just possibilities: CHIME is a generic architecture, no particular theme is built-in. But environment designers do not necessarily need to program since graphic textures and models can be supplied by third parties, and the specific layout and contents of a world are automatically generated according to an XML-based configuration. The environment designers must, of course, understand their backend repositories sufficiently to write the XML and corresponding processors, unless such meta-information is already supplied by the sources.

Columbia's *Workgroup Cache* system also draws from Prof. Kaiser's prior experience investigating process-centered environments. The early-90's Laputa extension to Oz employed information about the software process or workflow to determine which documents to prefetch for later work while disconnected from the network (e.g., using a laptop). For example, Laputa might fetch all documents necessary for the completion of a selected task, plus documents necessary for tasks expected to follow that task shortly thereafter in the process. Workgroup Cache similarly considers workflow semantics to predict future data needs, but extends beyond Laputa by including the work processes of multiple users, i.e., multiple participants in the workflow, in its document prefetch criteria. Workgroup Cache also introduces recommendations, or pushes, of shared documents under certain circumstances.

A Workgroup Cache system operates as a virtual intranet, providing possibly remote cache sharing to members of the same workgroup. Criteria are associated with each workgroup to pull documents from an individual member's cache or an outside information resource to the shared cache, or push from the shared cache to an individual cache or to a user's screen. These criteria can (in principle) leverage any knowledge available about the content and usage of documents as a basis for prediction of future accesses and/or recommendations. Cache pull, replacement, and push criteria might be based on software process or workflow routing among workgroup members, document access patterns of workgroup members, or with XML metadata associated with or

embedded in accessed documents. For example, if my supervisor keeps returning to such and such technical report over a recent time interval, or wrote it, then I might want to read it too. Criteria might be defined via simple filter rules, like Cisco firewalls or Web search engine queries, or via a very elaborate event/data pattern notation.

Workgroup membership can be determined in a number of ways: The users can be specified in advance, such as a software development team working closely together (although they might be physically dispersed). Or workgroups can be constituted and updated dynamically, say, by including users whose document accesses, or whose own home page links, match patterns associated with the workgroup. For instance, the amateur programmers actively working on the same subsystem of an open-source project like Linux might be automatically added to the corresponding subsystem's workgroup when they submit updates. Users may be members of multiple workgroups at the same time.

3. Implications for Future Research

The Worklets line of research is being continued in the DASADA program, for process-aware repair/reconfiguration of distributed component-based systems. A variant targeted to survivability of multi-level secure enterprise applications is supported by ONR in collaboration with NRL.

NSF has supported continued development of CHIME as a software development environment. More recently, CHIME's backend information contextualization model (not necessarily including the 3D user interface) is being poised as a foundation for Habitat systems of systems. Habitats, a potential area for future DARPA research, are defined as a dynamic mix of systems, assets, organizations, or individuals assembled to perform given functions and operating within a given situational context.

Workgroup Cache is being applied to variable-bandwidth synchronized video, via prefetching/sharing video segments, in a distance-learning experimental project supported by NSF. The workgroup context model also seems likely to be applicable to Habitats.

4. Software

Currently supported software available at http://www.psl.cs.columbia.edu/software/download/. Older unsupported software available at http://www.psl.cs.columbia.edu/old.html.

5. Publications

Wenke Lee and Gail E. Kaiser. Interfacing Oz with the PCTE OMS: A Case Study of Integrating a Legacy System with a Standard Object Management System. *Journal of*

Systems Integration, Kluwer Academic Publishers, 9(4):329-358, December 1999. http://www.psl.cs.columbia.edu/ftp/psl/pcte.tar.gz.

Jingshuang J. Yang and Gail E. Kaiser. JPernLite: Extensible Transaction Services for WWW. *IEEE Transactions on Knowledge and Data Engineering*, 11(4):639-657, Jul/Aug 1999. http://www.psl.cs.columbia.edu/ftp/psl/CUCS-009-98.pdf.gz. (Expansion of JPernLite: An Extensible Transaction Server for the World Wide Web, in *9th ACM Conference on Hypertext and Hypermedia*, June 1998, pp. 256-266.)

Gail E. Kaiser, Stephen E. Dossick, Wenyu Jiang, Jack Jingshuang Yang and Sonny Xi Ye. WWW-based Collaboration Environments with Distributed Tool Services. *World Wide Web*, Baltzer Science Publishers, 1:3-25, January 1998. http://www.psl.cs.columbia.edu/ftp/psl/CUCS-003-97.ps.gz. (Expansion of Gail E. Kaiser, Stephen E. Dossick, Wenyu Jiang and Jack Jingshuang Yang, An Architecture for WWW-based Hypercode Environments, in *19th International Conference on Software Engineering: Pulling Together*, May 1997, pp. 3-12.)

Israel Z. Ben-Shaul and Gail E. Kaiser. Federating Process-Centered Environments: the Oz Experience. *Journal of Automated Software Engineering*, Kluwer Academic Publishers, 5(1):97-132, January 1998. http://www.psl.cs.columbia.edu/ftp/psl/CUCS-006-97.ps.gz. (The issue was reprinted as a book, Elisabetta Di Nitto and Alfonso Fuggetta (eds.), *Process Technology*, Kluwer Academic Publishers, 1997.)

Stephen E. Dossick and Gail E. Kaiser. CHIME: A Metadata-Based Distributed Software Development Environment. *Joint Seventh European Software Engineering Conference and Seventh ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, September 1999, pp. 464-475. http://www.psl.cs.columbia.edu/ftp/psl/CUCS-006-99.zip.

Gail Kaiser, Christopher Vaill and Stephen Dossick. A Workgroup Model for Smart Pushing and Pulling. *Eighth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, June 1999, pp. 15-21. http://www.psl.cs.columbia.edu/ftp/psl/CUCS-012-99.pdf.

Jack Jingshuang , Gail Kaiser, Steve Dossick and Wenyu Jiang. Integrating Transaction Services into Web-based Software Development Environments. *First Asia Pacific Web Conference: Web Technologies and Applications*, Part I, Chapter 21, September, 1998, pp. 199-208. http://www.psl.cs.columbia.edu/papers/CUCS-008-98/.

Gail E. Kaiser and Stephen E. Dossick. Workgroup Middleware for Distributed Projects. *IEEE Seventh International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, June 1998, pp. 63-68. http://www.psl.cs.columbia.edu/ftp/psl/CUCS-006-98.ps.gz.

Wenyu Jiang, Gail E. Kaiser, Jack Jingshuang and Stephen E. Dossick. WebCity: A WWW-based Hypermedia Environment for Software Development. Poster paper in

Seventh Workshop on Information Technologies and Systems, December 1997, pp. 241-245. http://www.psl.cs.columbia.edu/ftp/psl/wits97.pdf.

Dan Port and Gail Kaiser. Introducing a "Street Fair" Open Source Practice Within Project Based Software Engineering Courses. Position paper in *First Workshop on Open Source Software Engineering*, May 2001. http://www.psl.cs.columbia.edu/ftp/psl/portkaiser.pdf.

Barry Boehm, Gail Kaiser and Daniel Port. A Combined Curriculum Research and Curriculum Development (CRCD) Approach to Software Engineering Education. Position paper in *Conference on Software Engineering Education and Training: Workshop on Developing Undergraduate Software Engineering Programs*, March 2000.

Gail Kaiser, Adam Stone and Stephen Dossick. A Mobile Agent Approach to Lightweight Process Workflow. Position paper in *International Process Technology Workshop*, September 1999. http://www.psl.cs.columbia.edu/ftp/psl/CUCS-021-99.pdf.

Stephen E. Dossick and Gail E. Kaiser. Distributed Software Development with CHIME. Position paper in *ICSE-99 Second Workshop on Software Engineering over the Internet*, May 1999. http://www.psl.cs.columbia.edu/papers/CUCS-007-99.html.

Gail Kaiser. From Oz To TreatyMaker. Position paper in *WACC '99 Workshop on Cross-Organisational Workflow Management and Co-ordination*, February 1999.

Daniel Port and Gail Kaiser. Collaborative Work: Collaborative Technologies for Evolving Software Systems. Column in *IEEE Internet Computing*, 2(6):79-83, November/December 1998.

Stephen E. Dossick and Gail E. Kaiser. Worklets for Adaptive Workflow. Position paper in *CSCW-98 Workshop: Towards Adaptive Workflow Systems*, November 1998.

Frank Maurer and Gail Kaiser. Software Engineering in the Internet Age. Guest Editors' Introduction in *IEEE Internet Computing*, 2(5):22-24, September/October 1998.

Shih-Fu Chang, Luis Gravano, Gail E. Kaiser, Kenneth Ross, Salvatore J. Stolfo. Database Research At Columbia University. Unrefereed article in *SIGMOD RECORD*, 27(3):75-80, September 1998.

Andrew P. Kosoresow and Gail E. Kaiser. Collaborative Work: Using Agents to Enable Collaborative Work. Column in *IEEE Internet Computing*, 2(4):85-87, July/August 1998.

Israel Ben-Shaul and Gail Kaiser. Collaborative Work: Coordinating Distributed Components over the Internet. Column in *IEEE Internet Computing*, 2(2):83-86, March/April 1998.

Peyman Oreizy and Gail Kaiser. Collaborative Work: The Web as Enabling Technology for Software Development and Distribution. Column in *IEEE Internet Computing*, 1(6):84-87, November/December 1997.

Stephen E. Dossick and Gail Kaiser. Collaborative Work: Tool Services for Intranets. Column in *IEEE Internet Computing*, 1(5):80-81, September/October 1997.

Roy T. Fielding and Gail Kaiser. Collaborative Work: The Apache HTTP Server Project. Column in *IEEE Internet Computing*, 1(4):88-90, July/August 1997.

Gail E. Kaiser, Stephen E. Dossick, Wenyu Jiang and Jack J. Yang. An Open Hypertext Collaboration Environment for the World Wide Web and Other Distributed Computing Infrastructures. Position paper in Uffe K. Wiil (ed.), *Third Workshop on Open Hypermedia Systems*, April 1997, pp. 86-92.

Gail Kaiser and Jim Whitehead. Collaborative Work: Distributed Authoring and Versioning. Column in *IEEE Internet Computing*, 1(2):76-77, March/April 1997.

6. Participants

Faculty: Gail Kaiser.

Graduate Research Assistants: Steve Dossick, Wenyu Jiang, Jack (Jingshuang) Yang, Sonny (Xi) Ye, David Orme, Xiaoning Zhang, Chris Vaill, Scott (Kai) Lei, Adam Stone, Janak Parekh, Peppo Valetto, Phil Gross, Gaurav Kc, Alpa Shah, Kanan Naik, Suhit Gupta, Denis Abramov.

P/T Research Staff: Scott Gross, Alex Voskoboynik, Justin Miranda, Frank (Jing) Fan.

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.